when they add/declare new scheme instances). Implementers can thus model device differences either via distinct facet schemes (i.e., by choosing to introduce distinct schemes to the schema) or via distinct parameter/property values (i.e., by changing logical description values for an existing scheme instance).

Physical Schema

[0080] Thus far, we have described the use of a declarative, logical description of a peripheral hardware environment. A corresponding physical description (of the peripheral hardware environment) is also needed. The physical description is paired with a corresponding logical description to complete a description.

[0081] Returning to the running power supply example, suppose that there is a specific type of power supply in use. To illustrate the variety of physical descriptions often required, suppose that this power supply is controlled through certain general purpose input/output pins (which are available on the given host chip/hardware). Further suppose that there is a second power supply dedicated to some always-on embedded manageability sub-system. Alternatively, a third power supply instance might be an uninterruptible power supply (e.g., driven by batteries).

[0082] Although this second power supply instance might be direct wired to the host (operating) platform, like the first one, it might be quite different. At the logical level, the second power supply might not handle any system (board) interrupts at all. It might have no knowledge of front-panel button push events, it might be connected to the host/operational platform using different wires/pins, and it might use different electrical signal pulse widths (and/or triggering conventions).

[0083] The single, fixed software/firmware image according to embodiments of the present invention can handle these physical configuration differences just as easily as it does any other differences within a given scheme. As with the scheme differences (discussed above), the present invention obviates any need to change the software/firmware image for these physical/wiring level differences. The present invention can even handle changing hardware environments (e.g., hot-swap) without any software/firmware image changes.

[0084] To deal with some of the Power Supply physical aspects stipulated above, the present invention associates each instance of a logical peripheral device (e.g., each power supply) with various features on (or accessed through) the operational/host hardware (e.g., a microcontroller core based Application Specific Integrated Circuit—ASIC). Some peripheral schemes, like the one for power supply, may have facets that need to be associated with ports or pins (or some such physically distinct bit of hardware). Other peripheral schemes, like the one for fans, need to be associated with fan tachometer circuits (for fan speed readings) and other bits of hardware (like pulse width modulator circuits used for fan speed control). In contrast to power supplies, fans associate neither with operational/host ports nor operational/host pins.

[0085] The present invention supports support such context sensitive associations (scheme-to-scheme dependency/delegation relationships) via the block of bits (described above) in order to keep the context sensitive details consistent across the entire (system) schema. As described above, this block of bits has an interpretation (i.e., a format) that can vary by the specific scheme (e.g., Power Supply) in question. The specific (prototype) scheme dictates that interface facets. Thus, it dictates just what the block of bits interpretation will be in this scheme context.

[0086] Unless the logic knows something about the current scheme, the block of bits is utterly opaque. Much of the code found in any embedded firmware/software image will manipulate any such block of bits as just another collection of bits. Only the scheme-specific logic, like the facet implementation logic, embeds any scheme specific knowledge. Only scheme-specific logic can correctly interpret the block of bits. This scheme specific information/knowledge hiding is important. It forces strong modularity/encapsulation boundaries around each (prototype) scheme within a (prototype) schema.

[0087] The block-of-bits values may also vary by scheme. This sort of scheme-specific value is called a reflection value. It reflects and/or represents information common to all instances of given scheme. A portion of every block of bits may be required to be consistent for all sub-schemes of some enclosing, more general scheme. This sort of recursively-nested, sub-scheme (prototype hierarchy) is considered a part of the present invention. Most often, the block of bits values vary for each specific scheme instance. These simply reflect the particular property/attribute values that best describe the scheme instance in question.

[0088] With reference again to the running power supply example, many power supplies will use certain bits, e.g., to identify which operational/host ports or pins they use. To keep the example simple, and without any limitation on the scope of the invention, we focus here on a scheme instance for a Power Supply that communicates via operational/host ASIC pins. Such pins carry what are often called general-purpose input/output (GPIO) signals. The subordinate scheme used for (logical) GPIO signals can describe both the classic single-bit GPIO pin and cases where the logical GPIO signal is delivered using multiple GPIO pins (as a crude form of a bus). Such collections of GPIO pins can manifest a so-called bit-banged bus. To keep this example simple, the GPIO Bus Flag is set to false. In other words, the example Power Supply (instance) will use a simple (logical) GPIO signal. This makes all of the GPIO bus related block of bits parameters moot (at least for this particular example Power Supply). All the same, these GPIO bus related parameters are shown (in Table 2 below) to illustrate one particular scheme that has resulted from the application of the present invention.

TABLE 2

Logical Description of General Purpose IO (GPIO) Signal #4
(Block of bits only)

Section/Bit Offset	# bits	Purpose	Sample Value
1	16	Packed Bit Field	Ox0060
15:8	8	Reserved (presently unused)	0x00
7:0	8	(Logical) GPIO Pin Number	0x60
2	16	Dedicated Bit Field	0x0060
15:0	16	GPIO Signal Mask	0x0060
3	32	Packed Bit Field	0x053205FF